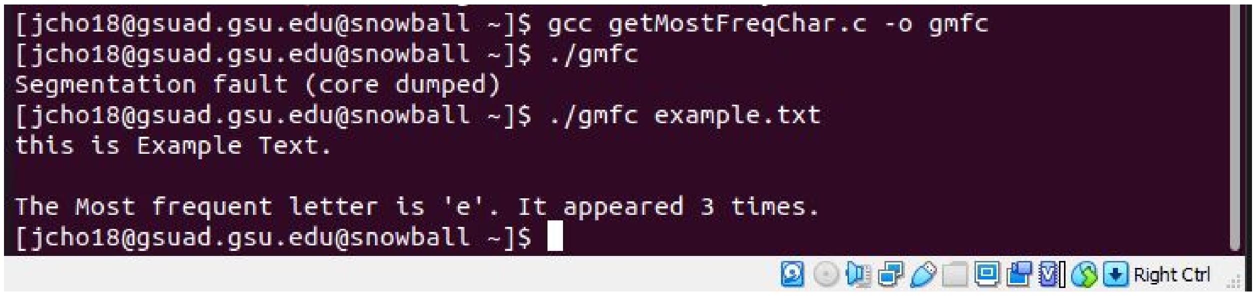
Part 1.

1.

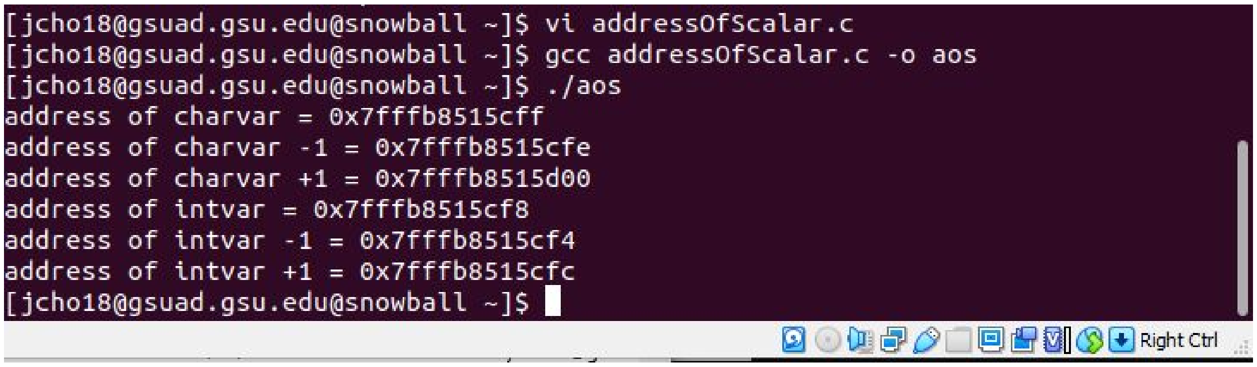


```
[jcho18@gsuad.gsu.edu@snowball ~]$ gcc getMostFreqChar.c -o gmfc
[jcho18@gsuad.gsu.edu@snowball ~]$ ./gmfc
Segmentation fault (core dumped)
[jcho18@gsuad.gsu.edu@snowball ~]$ ./gmfc example.txt
this is Example Text.

The Most frequent letter is 'e'. It appeared 3 times.
[jcho18@gsuad.gsu.edu@snowball ~]$
```

Part 2.

1.



```
[jcho18@gsuad.gsu.edu@snowball ~]$ vi addressOfScalar.c
[jcho18@gsuad.gsu.edu@snowball ~]$ gcc addressOfScalar.c -o aos
[jcho18@gsuad.gsu.edu@snowball ~]$ ./aos
address of charvar = 0x7fffb8515cff
address of charvar -1 = 0x7fffb8515cfe
address of charvar +1 = 0x7fffb8515d00
address of intvar = 0x7fffb8515cf8
address of intvar -1 = 0x7fffb8515cf4
address of intvar +1 = 0x7fffb8515cfc
[jcho18@gsuad.gsu.edu@snowball ~]$
```

2.

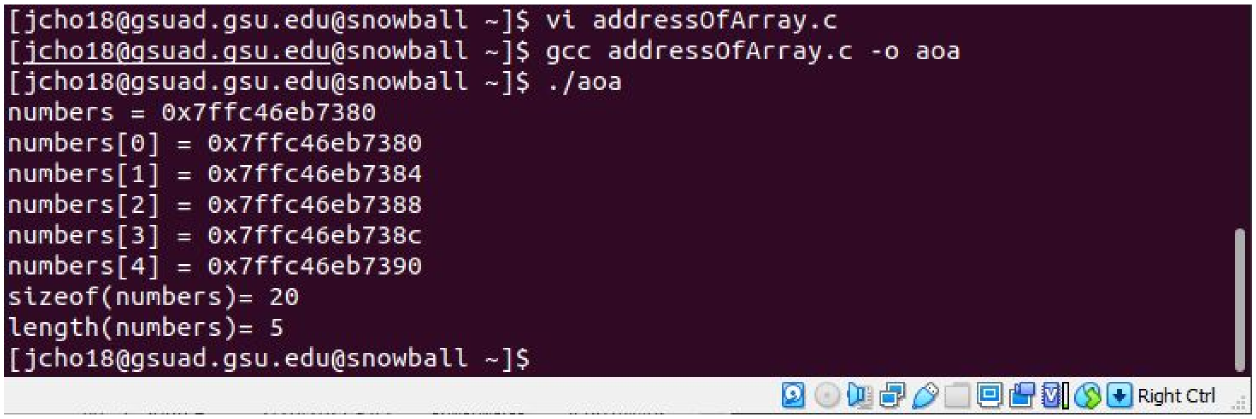
```
#include<stdio.h>
int main()
{
char charvar='\0';
printf("address of charvar = %p\n",(void*)(&charvar));
printf("address of charvar -1 = %p\n",(void*)(&charvar-1));
printf("address of charvar +1 = %p\n",(void*)(&charvar+1));
int intvar=1;
printf("address of intvar = %p\n",(void*)(&intvar));
printf("address of intvar -1 = %p\n",(void*)(&intvar-1));
printf("address of intvar +1 = %p\n",(void*)(&intvar+1));
}
```

3.

Intvar variable takes 4 bytes of memory, address increases by 4 bytes instead of 1.
Charvar variable takes 1 byte memory, address increases by 1 byte instead of 4.

Part 3.

1.



```
[jcho18@gsuad.gsu.edu@snowball ~]$ vi addressOfArray.c
[jcho18@gsuad.gsu.edu@snowball ~]$ gcc addressOfArray.c -o aoa
[jcho18@gsuad.gsu.edu@snowball ~]$ ./aoa
numbers = 0x7ffc46eb7380
numbers[0] = 0x7ffc46eb7380
numbers[1] = 0x7ffc46eb7384
numbers[2] = 0x7ffc46eb7388
numbers[3] = 0x7ffc46eb738c
numbers[4] = 0x7ffc46eb7390
sizeof(numbers)= 20
length(numbers)= 5
[jcho18@gsuad.gsu.edu@snowball ~]$
```

2.Yes, they are the same.

3. printf("length(numbers)= %lu\n", sizeof(numbers)/sizeof(numbers[0]));